Unsupervised Extremely Randomized Trees

Kevin Dalleau, Miguel Couceiro, Malika Smail-Tabbone

Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
kevin.dalleau@loria.fr;miguel.couceiro@loria.fr;malika.smail@loria.fr

Abstract. In this paper we present a method to compute dissimilarities on unlabeled data, based on extremely randomized trees. This method, Unsupervised Extremely Randomized Trees, is used jointly with a novel randomized labeling scheme we describe here, and that we call *AddCl3*. Unlike existing methods such as *AddCl1* and *AddCl2*, no synthetic instances are generated, thus avoiding an increase in the size of the dataset. The empirical study of this method shows that Unsupervised Extremely Randomized Trees with *AddCl3* provides competitive results regarding the quality of resulting clusterings, while clearly outperforming previous similar methods in terms of running time.

Keywords: Clustering, unsupervised classification,decision tree, extremely randomized trees, similarity measure, distance

1 Introduction and preliminaries

Many unsupervised learning algorithms rely on a metric to evaluate the pairwise distance between samples. Despite the large number of metrics already described in the literature [1], in many applications, the set of available metrics is reduced by intrinsic characteristics of the data and of the chosen algorithm. The choice of a metric may strongly impact the quality of the resulting clustering, thus making this choice rather critical.

Shi and Horvath [2] proposed a method to compute distances between instances in unsupervised settings using Random Forest (RF). RF [3] is a popular algorithm for supervised learning tasks, and has been used in various settings ([4, 5]). It is an ensemble method, combining decision trees in order to obtain better results in supervised learning tasks. Let $L = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set, where $X = \{x_1, \ldots, x_n\}$ is a list of *samples* (*i.e.*, feature vectors) and $Y = \{y_1, \ldots, y_n\}$ is the list of corresponding class labels. The algorithm begins by creating several new training sets, each one being a bootstrap sample of elements from X. A decision tree is built on each training set, using a sample of m_{try} features at each split. The prediction task is performed by performing a majority vote or by averaging the results of each tree, according to the problem at hand (classification or regression). This approach leads to better accuracy and generalization capacity of the model, and it reduces the variance of single decision trees [6].

The method proposed by Shi and Horvath, called Unsupervised RF (URF), derives from the common RF algorithm. Once the forest has been constructed, the training data can be run down each tree. Since each leaf only contains a small number of objects, and all objects of the same leaf can be considered similar, it is possible to define a similarity measure from these trees: if two objects i and j are in the same leaf of a tree, the overall similarity between the two objects is increased by one. This similarity is then normalized by dividing by the number of trees in the forests. In doing so, the similarities lie in the interval $[0, 1]$. The use of this RF is made possible in the unsupervised case thanks to the generation of synthetic instances, enabling binary classification between the latter and the observed instances. Two methods for data generation are presented in [2], namely, *addCl1* and *addCl2*.

In *addCl1*, the synthetic instances are obtained by a random sampling from the observed distributions of variables, whereas in *addCl2* they are obtained by a random sampling in the hyper-rectangle containing the observed instances. The authors found that *addCl1* usually leads to better results in practice. URF as a method for measuring dissimilarity presents several advantages. For instance, objects described by mixed types of variables as well as missing values can be handled. The method has already been successfully used in fields such as biology ([7–9]) and image processing [10].

However, the method, albeit its appealing character, suffers from some drawbacks. Firstly, the generation step is not computationally efficient. Since the obtained trees highly depend on the generated instances, it is necessary to construct many forests with different synthetic instances and average their results, leading to a computational burden. Secondly, the synthetic instances may bias the model being constructed to discriminate objects on specific features. For example, *addCl1* leads to forests that focus on correlated features.

P.Geurts, D.Ernst and L. Wehenkel [11] presented a novel type of ensemble of trees method that they called Extremely Randomized Trees (or ExtraTrees, for short). This algorithm is very similar to RF in many ways. In RF, both the instance and feature samplings are performed during the construction of each tree. In ExtraTrees (ET) another layer of randomization is added. Indeed, whereas in RF the threshold of a feature split is selected according to some purity measure (the most popular ones being the entropy and the Gini impurity), in ET these thresholds are obtained totally or partially at random. In addition, instead of growing the trees from bootstrapped samples of the data, ET uses the whole training set. At each node, K attributes are randomly selected and a random split is performed on each one of them. The best split is kept and used to grow the tree. The ET algorithm is described in Figure 1.

Two parameters are of importance in this algorithm: K, that we already defined, and n_{min}, that is the minimum sample size for a node to be split. Interestingly, the parameter K, that takes values in $\{1, ..., n_{features}\}$, influences the randomness of the trees. Indeed, for small values of K, the dependence of the constructed trees on the output variables gets weak. In the extreme case

Build_an_extra_tree_ensemble(S).
Input: a training set S.
Output: a tree ensemble $\mathcal{T} = \{t_1, \ldots, t_M\}$.
– For i=1 to M
  • Generate a tree: t_i= **Build_an_extra_tree**(S);
– Return $\mathcal{T}$.

Build_an_extra_tree(S).
Input: a training set S.
Output: a tree t.
– Return a leaf labeled by class frequencies (or average output, in regression) in S if
  (i) $|S| < n_{min}$, or
  (ii) all candidate attributes are constant in S, or
  (iii) the output variable is constant in S
– Otherwise:
  1. Select randomly K attributes, $\{a_1, \ldots, a_K\}$, without replacement, among all (non constant in S) candidate attributes;
  2. Generate K splits $\{s_1, \ldots, s_K\}$, where s_i = **Pick_a_random_split**(S, a_i), $\forall i = 1, \ldots, K$;
  3. Select a split s_* such that Score(s_*, S) = $\max_{i=1,\ldots,K}$ Score(s_i, S);
  4. Split S into subsets S_l and S_r according to the test s_*;
  5. Build t_l = **Build_an_extra_tree**(S_l) and t_r = **Build_an_extra_tree**(S_r) from these subsets;
  6. Create a node with the split s_*, attach t_l and t_r as left and right subtrees of this node and return the resulting tree t.

Pick_a_random_split(S,a)
Input: a training set S and an attribute a.
Output: a split.
– If the attribute a is numerical:
  • Compute the maximal and minimal value of a in S, denoted respectively by $a^S_{\min}$ and $a^S_{\max}$;
  • Draw a cut-point a_c uniformly in $[a^S_{\min}, a^S_{\max}]$;
  • Return the split $[a < a_c]$.
– If the attribute a is categorical (denote by $\mathcal{A}$ its set of possible values):
  • Compute $\mathcal{A}_S$ the subset of $\mathcal{A}$ of values of a that appear in S;
  • Randomly draw a proper non empty subset $\mathcal{A}_1$ of $\mathcal{A}_S$ and a subset $\mathcal{A}_2$ of $\mathcal{A} \backslash \mathcal{A}_S$;
  • Return the split $[a \in \mathcal{A}_1 \cup \mathcal{A}_2]$.

Fig. 1. The figure presets the ExtraTrees algorithm as extracted from [11]

where K is set to 1 (*i.e.*, only one feature is selected and randomly split), the dependence of the trees on the observed label is removed.

Following the tracks of [2] on URF, we propose to use ET with a novel approach where synthetic case generation is no longer necessary. This approach, that we call *addCl3*, consists of a random labeling of each instance. Using properties of ET that we will discuss below, it is possible to compute a good similarity measure from a dataset where *addCl3* is applied. The method outperforms URF in running time, while giving similar or better clusters.

This paper is organized as follows. After a description of the method in Section 2, we focus on the empirical evaluation on real-world datasets in Section 3, before reviewing of the method and giving some perspectives in Section 4.

2 Unsupervised Extremely Randomized Trees

Two methods are used for the generation of synthetic data: *addCl1* and *addCl2*. In these methods, the generation consists in sampling in the observed data. The synthetic data is assigned a label, while the observed data is assigned another one, enabling binary classification between observed and synthetic examples. The novel method we propose and evaluate in this work that we refer to as *addCl3*, does not focus on the generation of synthetic instances, but on the generation of labels instead. In *addCl3*, the label generation runs as follows:

1. Let n_{obs} be the number of instances in the dataset. A list containing $\lfloor \frac{n_{obs}}{2} \rfloor$ times the label 0 and $n_{obs} - \lfloor \frac{n_{obs}}{2} \rfloor$ times the label 1 is generated.
2. For each instance in the dataset, a label is randomly sampled without replacement from the aforementioned list.

This procedure ensures that the label distribution is balanced in the dataset. However, this leads to the same problem arising with *addCl1* and *addCl2*: the results are highly dependent on the generation step, as different realizations of the instance-label association or of the synthetic data may lead to completely different forests. To circumvent this issue, one solution is to run multiple forests on multiple generated datasets, and to average the results. Shi and Horvath found out that averaging the results from 5 forests, with a total of 5000 trees leads to robust results. Moreover, instead of running multiple forests on many generated datasets, it may be possible - and computationally more efficient - to run a single forest with a large amount of generated data, if some care is taken regarding the reweighting of each class. This workaround, proposed by a reviewer in [2], is easier to implement when our generation scheme is used. Indeed, since we do not add new instances, it is not necessary to reweight each class. Instead, we propose to duplicate the original dataset multiple times and apply *addCl3* to obtain a balanced dataset. This approach is evaluated in Section 3.

With *addCl3*, the construction of the trees no longer depends on the structure of the data. Indeed, when *addCl1* or *addCl2* are used, the forests are trained to distinguish between observed and synthetic instances. In *addCl3*, the labels being assigned randomly, two similar instances may be labeled differently and may fall in different leaves. However, using ET with the number of features randomly selected at each node $K = 1$, the construction of the trees no longer depends on the class label, as described in the previous section. Hence, ET seems to be a suitable algorithm to use with *addCl3*. Algorithm 1 describes the Unsupervised Extremely Randomized Trees (UET) method.

Algorithm 1: Unsupervised Extremely Randomized Trees

Data: Observations O
Result: Similarity matrix S
$D \longleftarrow addCl3(O)$;
$T \longleftarrow Build\_an\_extra\_tree\_ensemble(D, K)$ // *Here* $K = 1$;
$S = 0_{n_{obs}, n_{obs}}$ // *Initialization of a zero matrix of size* n_{obs} ;
for $d_i \in D$ **do**
  for $d_j \in D$ **do**
    $S_{i,j}$ = number of times the samples d_i and d_j fall in the same leaf node in each tree of T $= \{t_1, t_2, ..., t_M\}$;
  end
end
$S_{i,j} = \frac{S_{i,j}}{M}$;

The algorithm $Build\_an\_extra\_tree\_ensemble(D)$ is given in Figure 1. A few parameters can influence the results of UET:

1. The number of copies of the original dataset n_{copies} before applying $addCl3$.
2. The number of trees n_{trees}.
3. The minimum number of samples for a node to be split n_{min}.

3 Empirical evaluation

In this section, we investigate the influence of the parameters introduced at the end of Section 2, as well as the performance of the method.

3.1 Optimization of parameters

For each evaluation presented in this subsection, the following process is repeated 10 times:

1. A similarity matrix is constructed using UET.
2. This similarity matrix is transformed into a distance matrix using the relation $DIS_{ij} = \sqrt{1 - SIM_{ij}}$, used in [2].

3. An agglomerative clustering (with average linkage) is performed using this distance matrix, with the relevant number of clusters for the dataset.

For each clustering, Adjusted Rand Indices (ARI) are computed. This measure quantifies the agreement between two partitions of a dataset, adjusted for chance [12, 13]. ARI takes values in $[-1, 1]$, where a value of 1 indicates perfect agreement up to a permutation, while a value of 0 indicates a result no better than a random label assignment.

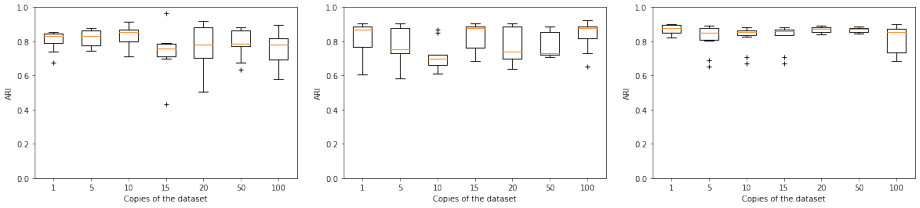
Three datasets are used for this evaluation process: Iris [14], Wine [15] and Wisconsin breast cancer [16]. These datasets are described Table 1.

Table 1. Properties of used datasets

Dataset	# samples	# features	# labels
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2

Influence of the number of copies of the dataset

The use of *addCl*3 on a dataset leads to a balanced distribution of the labels. Instead of running k forests on as many datasets where *addCl3* is applied k times, it is possible to run one forest on a dataset duplicated k times. We evaluate here the influence of this duplication process. The results are presented Figure 2.



(a) (b) (c)

Fig. 2. ARI performing UET+*addCl3* and agglomerative clustering on Wine (a), Iris (b) and Wisconsin (c) datasets when the number of copies of the dataset increases.

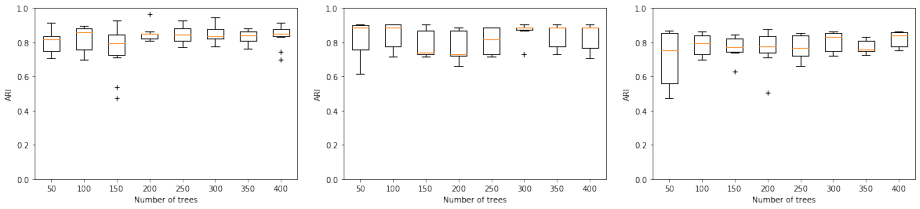
The ARI are compared using the Kruskal-Wallis test [17]. The results show that the ARI does not differ significantly in Wine, Iris and Wisconsin datasets ($p = 0.26$, $p = 0.09$ and $p = 0.23$, respectively).

Intuitively, as UET grows the tree without any consideration of the labels and without bootstrapping the samples, the results should stay relatively constant

when number of duplications grows. This replication was needed in URF as the generation scheme could lead to significant differences in the output similarity. This intuition is confirmed with this experiment. Here, the randomness induced by the labeling step does not induce a difference in the construction of the trees. Indeed, since we set $K = 1$, trees are constructed totally at random. Any difference in the similarity matrix is rather related to the randomness induced by the choice of features to split at each node.

Influence of the number of trees

The influence of the number of trees n_{trees} as also been studied in [11], where this parameter is referred to as the averaging strength M. For randomized method such as RF and ET used in a supervised learning setting, the average error is a monotonically decreasing function of M [3]. In our experiments, we observed no substantial gain for $n_{trees} > 50$. The difference in ARI are not significant ($p > 0.1$) for all three datasets. This observation confirms the one from Geurts *et al.* where values of $n_{trees} > 40$ outperforms Tree Bagging. However, as the time to construct the ensemble grows linearly with the number of trees, it is a good option to choose small a value of n_{trees}. We chose the value $n_{trees} = 50$ by default. We noticed that this value is way below the overall number of trees recommended for URF, 5000. The results are presented Figure 3.



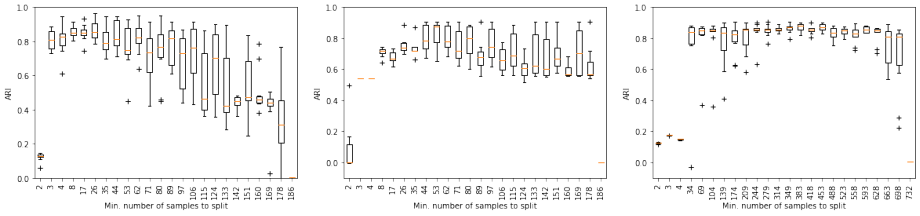
(a) (b) (c)

Fig. 3. ARI performing UET+*addCl3* and agglomerative clustering on Wine (a), Iris (b) and Wisconsin (c) datasets when the total number of trees varies. The ARI remains relatively constant.

Influence of the minimum number of samples to split

ET tend to produce trees having 3 to 4 times the number of leaves than those of RF. As UET computes similarities by counting the number of times objects fall into the same leaf, the results are impacted by this increase in the number of leaves. It might be useful to stop the growth of the trees, in order to group

similar instances in the same leaves more often. The minimum number of objects to split a node n_{min} can control this growth. This parameter n_{min}, also called the *smoothing strength*, has an impact on the bias and the variance. As stated by Geurts *et al.* [11], the optimal value for this parameter depends on the level of noise in the dataset. They showed in [11] that larger values of n_{min} are needed when ET is applied to noisy data. In UET, the noise is maximal, as the labels are assigned randomly. The results of the evaluations performed varying n_{min} are presented below. For $n_{min} = 2$, we observe that the method fails to compute a similarity matrix leading to a good clustering. Values of n_{min} between 20% and 30% of the data seem to give better results. The ARI variations for the three datasets according to n_{min} are presented Figure 4.



(a) (b) (c)

Fig. 4. ARI performing UET+*addCl3* and agglomerative clustering on Wine (a), Iris (b) and Wisconsin (c) datasets when the min. number of samples to split increases. Last value corresponds to 110% of the samples in a dataset.

3.2 Comparative evaluation of UET

In this section we first evaluate the relevance of clusterings obtained using UET by comparing the Normalized Mutual Information [18] (NMI) scores with the values presented in [19]. This reference was chosen because results were provided for many well-known datasets. Then, we compare UET and URF, using another quality score presented previously, ARI. The ten datasets used in this section are available on the UCI website [1] and presented Table 2. UET are computed with $n_{trees} = 50$ and $n_{min} = \lceil \frac{n_{samples}}{3} \rceil$.

Comparative evaluation with results from the literature

For each dataset, UET was run 10 times, and the similarity matrices were averaged. The obtained matrix was then transformed into a distance matrix using the equation $DIS_{ij} = \sqrt{1 - SIM_{ij}}$, and an agglomerative clustering with

[1] https://archive.ics.uci.edu/ml/index.php

Table 2. Datasets used for benchmarking

Dataset	# samples	# features	# labels
Iris	150	4	3
Wine	178	13	3
Wisconsin	699	9	2
Lung	32	56	3
Breast tissue	106	9	6
Isolet	1559	617	26
Pima	768	8	2
Parkinson	195	22	2
Ionosphere	351	34	2
Segmentation	2310	19	7

the relevant number of clusters was performed. The quality of the clustering was then evaluated with respect to NMI. This process is run 20 times, and we provide the mean and standard deviation of the quality metric. This evaluation was performed using *scikit-learn* [20] and our implementation of UET. This implementation will be available upon request.

In [19], NMI obtained by running k-means 20 times and averaging the results are provided for each dataset. We compare our results to the ones obtained without feature selection, as none has been performed in our setting. The results are presented Table 3. They show that NMI scores obtained using UET are competitive in most cases. It is noteworthy that in some cases, UET alone without feature selection gives better results than the ones obtained by [19] after feature selection. For instance, this is the case for *Breast tissue* dataset.

Table 3. Comparative evaluation with the results from [19]. Best obtained values are indicated in boldface. In case of a tie, both values are in boldface. Time comparison was not performed in this case.

Dataset	UET - NMI	Literature - NMI
Wisconsin	**72.95** ± 4.94	**73.61** ± 0.00
Lung	**28.89** ± 5.76	22.51 ± 5.58
Breast tissue	**59.59** ± 1.03	51.18 ± 1.38
Isolet	**69.95** ± 1.20	69.83 ± 1.74
Parkinson	**21.06** ± 5.33	**23.35** ± 0.19
Ionosphere	**13.48** ± 3.25	**12.62** ± 2.37
Segmentation	**69.31** ± 1.51	60.73 ± 1.71

Comparison with URF

To compare UET and URF, we used the R implementation provided by Shi and Horvath [2], and compared the ARI obtained after running the partitioning around medoids (PAM) algorithm on the distance matrices obtained by both methods. 2000 trees and 100 forests are used for URF, with a value of $m_{try} = \lfloor \sqrt{n_{features}} \rfloor$ [3]. We set UET parameters to $n_{trees} = 50$ and $n_{min} = \lceil \frac{n_{samples}}{3} \rceil$, and averaged the similarity matrices of 20 runs. These experiments were run on a computer with an Intel i7-6600U (2.6 Ghz) and 16 Go of 2133 MHz DDR4 RAM.

We compared both ARI and time (in seconds) for each method. The results are presented Table 4. UET outperforms URF time-wise, while giving similar or better clusterings. Regarding the Isolet dataset, we manually terminated URF's computation as we weren't able to obtain results in an acceptable amount of time on our machine. However, we performed the computation on a more powerful machine, and were able to obtain an ARI of 28.39.

Table 4. Comparative evaluation between URF and UET

Dataset	UET (ARI - Time (s))	URF (ARI - Time (s))
Wisconsin	79.30 - **823.36 s**	**81.36** - 1267.82 s
Lung	**10.81 - 7.45 s**	8.16 - 89.32 s
Breast tissue	**40.35 - 25.25 s**	39.05 - 94.55 s
Isolet	**33.44 - 4589.36 s**	* - * s
Parkinson	**17.37 - 66.91 s**	13.44 - 252.12 s
Ionosphere	**8.54 - 184.97 s**	7.59 - 722.92 s

4 Conclusion and perspectives

In this preliminary work, we presented a novel method to perform unsupervised clustering using decision trees. This approach extends the unsupervised random forest method, by using extremely randomized trees as a base estimator. In the former method, the generation of synthetic instances was needed. This generation can be performed by two different approaches, *AddCl1* or *AddCl2*. With the approach we proposed here, the generation of instances is no longer necessary. Indeed, for some parameter choices, extremely randomized trees can be made independent of the labels. We therefore present a way to bypass the need for instance generation, *AddCl3*, where a label is randomly associated with each observation, which results in a significant reduction in running time.

A performance evaluation of our method showed that essentially one parameter influenced the results, the *smoothing parameter* n_{min}. This is explained by the fact that higher values of n_{min} give better results in the presence of noise. In our

[2] https://labs.genetics.ucla.edu/horvath/RFclustering/RFclustering.htm
[3] m_{try} is the number of variables used at each node when a tree is grown in RF.

case, the data is highly noisy, as the labeling is a random process. We found that a value of $\frac{n_{samples}}{4} \leq n_{min} \leq \frac{n_{samples}}{3}$ gives good clusterings. Other parameters, such as the number of trees per forest n_{trees} did not influence much the results of the procedure for values of $n_{trees} > 50$, while increasing the time to perform the procedure. An interesting finding is that it is no longer necessary to duplicate the dataset multiple times to improve the results. However, due to the randomness of the procedure, it is still necessary to average the results of multiple UET to decrease the variance.

We compared the quality of the clustering between our method and (i) results found in the literature and (ii) results obtained by URF on multiple datasets. The quality is measured by normalized mutual information or adjusted rand index, according to the metric available in the literature. This empirical evaluation gave promising results, with overall similar or better NMI and ARI. The advantages of our method over URF are twofold. First, the generation of synthetic data is no longer necessary. Second, the method is 1.5 to more than 10 times faster than URF.

However, there is still room for improvement. We are aware that we only have tested our method on a few small datasets so far. A comparison with other metrics on large synthetic and real-world datasets would be interesting. Moreover, one of the major advantages of using a decision tree-based method to compute a distance is that (i) it enables the use of mixed-type variables and (ii) missing data can be handled. In fact, the latter were our original motivation and they constitute topics for future work.

Acknowledgements Kevin Dalleau's PhD is funded by the RHU FIGHT-HF (ANR-15-RHUS-0004) and the Region Grand Est (France).

References

1. M. M. Deza and E. Deza. Encyclopedia of distances. In *Encyclopedia of Distances*, pages 1–583. Springer, 2009.
2. T. Shi and S. Horvath. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138, 2006.
3. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
4. B. Percha, Y. Garten, and R. B. Altman. Discovery and explanation of drug-drug interactions via text mining. In *Pacific Symposium on Biocomputing*, pages 410–421, 2012.
5. M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.
6. J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
7. H. L. Kim, D. Seligson, X. Liu, N. Janzen, M.H. Bui, H. Yu, T. Shi, A. S. Belldegrun, S. Horvath, and R.A. Figlin. Using tumor markers to predict the survival of patients with metastatic renal cell carcinoma. *The Journal of urology*, 173(5):1496–1501, 2005.

8. M. C. Abba, H. Sun, K. A. Hawkins, J. A. Drake, Y. Hu, M. I. Nunez, S. Gaddis, T. Shi, S. Horvath, and A. Sahin, *et al.* Breast cancer molecular signatures as determined by sage: correlation with lymph node status. *Molecular Cancer Research*, 5(9):881–890, 2007.
9. S. I. Rennard, N. Locantore, B. Delafont, R. Tal-Singer, E. K. Silverman, J. Vestbo, B. E. Miller, P. Bakke, B. Celli, and P.M. Calverley, *et al.* Identification of five chronic obstructive pulmonary disease subgroups with different prognoses in the eclipse cohort using cluster analysis. *Annals of the American Thoracic Society*, 12(3):303–312, 2015.
10. K.Y. Peerbhay, O. Mutanga, and R. Ismail. Random forests unsupervised classification: The detection and mapping of solanum mauritianum infestations in plantation forestry using hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):3107–3122, 2015.
11. P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
12. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.
13. L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
14. R.A. Fisher and M. Marshall. Iris data set. *RA Fisher, UC Irvine Machine Learning Repository*, 1936.
15. M. Forina, *et al.* An extendible package for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analysis and Technologies*, 16147, 1991.
16. O.L. Mangasarian and W.H. Wolberg. Cancer diagnosis via linear programming. *Computer Sciences Department, University of Wisconsin-Madison*, 1990.
17. W.H. Kruskal and W.A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.
18. A. Strehl and J. Ghosh. Cluster ensembles-a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3:583–617, 2002.
19. H. Elghazel and A. Aussem. Feature selection for unsupervised learning using random cluster ensembles. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 168–175. IEEE, 2010.
20. F. Pedregosa, *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.